BILLIAU Pierre

2ème Année Télécommunications

RAPPORT DE STAGE

Intitulé : Implantation d'une méthode de guidage pour véhicule de type voiture en C++.



Réalisé à l'INRIA Rhône-Alpes du 9 juillet au 21 septembre 2001, au sein du projet Sharp.
Responsable : M. Thierry Fraichard

Table des matières

Abstract

My internship of 11 weeks at INRIA (National Institute of Research in Computer science and Automation, Rhône-Alpes) was dedicated to work on the improvement of a steering method for car-like vehicles. A steering method is an algorithm that computes paths between two configurations without the presence of obstacles. The modelling of the problem also imposes other constraints, especially the continuity of the curvature. There were already steering methods that had been developed but I had to extend them or to define new ones. My work was divided into four parts:

1. The extension of the steering method 'CC-Reeds and Shepp': the method was still implemented but we had to extend it to compute shorter paths.
2. The computation of topological paths: it allows the steering method to be useful with the computation of the paths in the presence of obstacles.
3. The implementation of statistics functions: it allows the program user to compare the relative performances of the different steering methods.
4. The designing of a graphic user interface (GUI) which allows the user to test easily the steering methods.

I. Introduction

Ce rapport de stage présente le travail que j'ai réalisé pour valider ma deuxième année d'études d'ingénieur au Département Télécommunications de l'INPG (ENSIMAG/ENSERG). Il présente d'abord succinctement l'institut de recherches INRIA qui m'a accueilli, ainsi que l'équipe du projet SHARP qui m'a entouré. Il précise ensuite quel est le sujet du stage ainsi que la théorie et les travaux de recherche menés dans ce domaine. Il est pour moi nécessaire de définir dans ce rapport certaines notions et d'énoncer quelques résultats théoriques si l'on veut comprendre mon travail. J'ai essayé d'être bref sur la théorie, tout en essayant d'être le plus complet possible sur les travaux de recherche sur la planification de mouvement. A cet égard, si l'on veut plus de précisions sur un point théorique, on pourra se référer aux documents détaillés dont les références sont fournies dans la bibliographie. Enfin, j'ai essayé d'expliquer quel a été mon travail et quels ont été les enseignements bénéfiques de ce stage.

Je n'ai pas jugé pertinent de fournir les sources du code que j'ai produit pendant le stage. En effet, ce rapport ne constitue pas une documentation technique pour un programmeur qui désirerait comprendre ou reprendre mon travail (je le renvoie alors à la documentation de mon code), mais simplement une description détaillée de ce que j'ai réalisé ; c'est pourquoi je ne m'attarde pas sur les détails des aspects techniques.

Je tiens à remercier M. Thierry Fraichard pour m'avoir accepté en stage et pour m'avoir conseillé tout au long de mon travail, ainsi que l'INRIA pour m'avoir accepté en stage dans ses locaux.

II. Contexte du stage

Le stage s'est déroulé à l'INRIA Rhône-Alpes (Montbonnot) du 9 juillet au 21 septembre 2001, au sein du projet Sharp.

1. Présentation de l'INRIA

« Un institut de recherche au cœur de la société de l'information ».
L'INRIA est un établissement public à caractère scientifique et technologique qui mène des recherches avancées dans le domaine des sciences et technologies de l'information et de la communication. Ce domaine inclut l'informatique et l'automatique, mais aussi les télécommunications et le multimédia, la robotique, le traitement du signal et le calcul scientifique.



Bâtiment de l'INRIA

2. La recherche à l'INRIA

Les recherches à l'INRIA sont menées au sein de groupes de travail appelés des 'projets de recherche'. Un projet de recherche se caractérise par :

•une équipe d'une quinzaine de scientifiques.
•une unité thématique forte.
•un programme de travail et des objectifs à moyen terme.
•une autonomie financière et scientifique.
•des liens et des coopérations avec les partenaires industriels et scientifiques, en France et dans le monde.
•une évaluation rigoureuse des résultats.
•une durée limitée.

Pour mieux cerner quelles sont les activités précises de l'INRIA, il peut être utile de présenter quels sont les différents projets de recherche menés dans l'établissement.

3. Les différents projets de recherche à l'INRIA

Maîtriser les réseaux et systèmes informatiques
Réseaux, parallélisme et systèmes répartis
•APACHE
Algorithmique, programmation parallèle et partage de charge
•ARENAIRE
Arithmétique des ordinateurs
•PLANETE
Protocoles et applications pour l'Internet
•ReMaP
Régularité et parallélisme massif
•RESO
Réseaux haut débit et applications
•SIRAC
Systèmes informatiques répartis pour applications coopératives
•VASY
Validation de systèmes
Aider à la conception et à la création
Bases de connaissances, documents multimédia, modèles cognitifs
•EIFFEL
Cognition et coopération en conception
•EXMO
Echanges de connaissance structurée médiatisés par ordinateur
•HELIX
Informatique et génomique
•OPERA
Outils pour les documents électroniques : recherche et applications
Percevoir, simuler et agir
Synthèse d'images, réalité virtuelle, vision par ordinateur et robotique
•iMAGIS
Modèles, algorithmes, géométrie pour le graphique et l'image de synthèse
•MOVI
Modélisation, localisation, reconnaissance et interprétation en vision
•SHARP
Programmation automatique et systèmes décisionnels en robotique
Modéliser les phénomènes complexes
Automatique, simulation et calcul scientifique
•BIP
Robot bipède, contrôle commande et programmation temps-réel
•IDOPT
Identification et optimisation de systèmes en physique et en environnement
•IS2
Inférence statistique pour l'industrie et la santé
•NUMOPT
Optimisation numérique
•SINUS
Simulation numérique dans les sciences de l'ingénieur

4. Le projet Sharp

Mon stage s'est déroulé au sein du projet Sharp (Programmation automatique et systèmes décisionnels en robotique).

i. Présentation du projet

Le projet SHARP centre son activité de recherche sur l'étude des problèmes liés à la modélisation et à la génération automatique du mouvement et des interactions physiques en robotique. Le terme « robotique » revêt ici un caractère particulier, dans le sens où il inclut à la fois des machines physiques (communément appelées «robots») capables d'actions autonomes dans le monde réel, et des agents mobiles ou articulés (ou « robots virtuels ») possédant des capacités de mouvements propres leurs permettant d'évoluer de manière autonome (ou semi-autonome) dans un monde virtuel possédant des lois physiques semblables à celles du monde réel.
SHARP est un projet commun avec le CNRS, l'INPG et l'Université Joseph Fourier.

ii. Axes de recherche

•Algorithmique pour la planification de mouvements (prise en compte de contraintes de non-collision, contraintes cinématiques et dynamiques, incertitude) dans des mondes réels ou virtuels ;
•Méthodologie pour le développement d'architectures décisionnelles pour le contrôle de robots mobiles dans des environnements dynamiques peu ou pas connus a priori ;
•Modèles et algorithmes pour la simulation dynamique, i.e. la gestion des interactions physiques et la simulation de la dynamique des corps complexes en mouvement et en interaction (déformations, collisions, forces...) ;
•Outils de modélisation et de calcul probabiliste pour la géométrie, permettant de traiter correctement les incertitudes et leurs impacts sur les problèmes inverses et les problèmes d'interprétation de données sensorielles que nous rencontrons (sujet traité en collaboration avec le laboratoire Leibniz de l'IMAG).

iii. Applications

L'activité de recherche précédente est à la fois valorisée et fertilisée par des activités plus appliquées qui visent au développement de solutions à des problèmes industriels. Plusieurs prototypes de recherche et expérimentations réelles (e.g. sur des véhicules autonomes ou des systèmes de vidéo professionnelle et de production d'images) sont ainsi réalisés en relation avec les moyens robotiques de l'INRIA Rhône-Alpes et des industriels ; certains de ces prototypes ont déjà donné lieu à des transferts de technologies (en CAO-robotique et en vidéo professionnelle en particulier).

Les applications plus particulièrement visées par cette activité de recherche sont celles de la robotique non manufacturière (e.g. maintenance d'équipements ou intervention en milieu hostile ou lointain, robotique de service...), en mettant l'accent sur les domaines du transport et du médical ; l'autre secteur d'application concerné par nos travaux sur le mouvement dans le monde virtuel est celui de la réalité virtuelle et du multimédia.

iv. Personnel

Il se compose de chercheurs à temps plein, mais aussi de thésards et de stagiaires. Le responsable scientifique du projet est M. Christian Laugier.

III. Sujet du stage

1. Description du projet

Le but du stage est de travailler sur *l'implantation d'une méthode de guidage pour véhicule de type voiture*. Ce travail de recherche entre dans le cadre beaucoup plus général du projet Sharp, qui lui-même entre dans le cadre du projet français *«La Route Automatisée »*. Le but de ce projet est la construction de véhicules autonomes. Dans ce cadre, de nombreux travaux de recherche sont effectués sur la **planification de mouvement**, c'est-à-dire la détermination du chemin que va emprunter un véhicule en fonction des paramètres internes au véhicule mais aussi extérieurs (par exemple les obstacles). Les paramètres internes sont les paramètres cinématiques du véhicule (position, vitesse), mais on tient aussi compte de certaines contraintes intrinsèques au véhicule, notamment la continuité de la courbure des véhicules, ainsi que de la valeur maximale de la courbure. On cherche en effet à modéliser de façon la plus proche de la réalité le mouvement des véhicules. Cette modélisation doit donc tenir compte de l'incapacité des roues d'un véhicule à tourner instantanément (ou alors le véhicule devrait stopper son mouvement, braquer les roues, puis redémarrer ; ce qui n'est pas très efficace...) et de la valeur maximale de la courbure des roues, c'est-à-dire que le rayon de courbure du véhicule dans un virage est limité. Nous verrons que ces contraintes sont fondamentales dans la planification de mouvement. Si le véhicule a la capacité de se déplacer en marche arrière, on doit en tenir compte pour pouvoir optimiser la longueur du chemin. Il y a aussi des contraintes extérieures à prendre en compte, notamment l'évitement des obstacles. Ces obstacles peuvent être immobiles mais aussi mobiles, par exemple d'autres véhicules sur une même route.

La planification de mouvement va donc déterminer une trajectoire réalisable par le véhicule en fonction de tous ces paramètres et d'autres critères, comme le compromis entre la longueur du chemin (qu'on tentera bien entendu de minimiser), et le temps de calcul du chemin, qu'on essaiera aussi de minimiser pour pouvoir développer l'application en temps réel sur de vrais véhicules.

2. Lien avec le stage

Le stage consiste à développer une **méthode de guidage** (“steering method ”), *i.e.* un algorithme qui calcule un chemin entre deux configurations du véhicule sans prendre en compte la présence d’obstacles dans l’environnement. Etant donnée une méthode de guidage, il est alors possible de déterminer le chemin global en utilisant un algorithme général de planification des chemins prenant en compte les obstacles (il en existe plusieurs connus : *the Probabilistic Path Planner, the Ariadne’s Clew Algorithm, the Holonomic Path Approximation Algorithm*). Une méthode de guidage est donc un élément clé dans l’algorithme général de planification de mouvement. Comme dit précédemment, on peut développer plusieurs méthodes de guidage, suivant les critères du résultat que l’on veut obtenir. Les méthodes de guidage développées pendant le stage devront tenir compte des contraintes du véhicule énoncées précédemment (continuité de la courbure dans les virages, limitations mécaniques du véhicule : rayon de braquage limité, variation de l’angle de braquage limitée).
Quelles seront les conséquences de ces contraintes sur l’algorithme à développer ?
Modélisons le problème puis examinons ces conséquences.

3. Modélisation du problème

Je vais ici présenter quelques résultats théoriques simples qui permettront de mieux comprendre quel était le contexte de mon stage. Considérons un véhicule à quatre roues (deux roues arrière et deux roues avant motrices). Il faut trois paramètres pour déterminer la position et l'orientation du véhicule et un paramètre pour déterminer l'orientation des roues avant. On définit la **configuration** du véhicule par un quadruplet $q = (x, y, \theta, \kappa)$, où (x, y) représente les coordonnées du milieu du segment joignant les roues avant, θ l'orientation du véhicule et κ l'orientation des roues avant. On définit un **chemin** comme un ensemble de fonctions continues $(x(t), y(t), \theta(t), \kappa(t))$. A partir de ce modèle, on peut déterminer quels sont les chemins admissibles, *i.e.* les chemins effectivement réalisables (un chemin admissible est solution d'un système d'un système différentiel que nous ne mentionnons pas ici pour simplifier). Cependant les chemins admissibles doivent vérifier les deux contraintes suivantes : la condition de roulement sans glissement (les roues tournent perpendiculairement à leur axe) et les contraintes sur la courbure.
Un chemin admissible est donc un chemin qui relie les configurations initiale et finale.
Le problème est donc de déterminer un chemin admissible à partir d'une configuration initiale q_s et d'une configuration finale q_g données, tel que :

- Il doit relier q_s et q_g.
- La longueur du chemin doit être minimale, si l'on trouve plusieurs chemins admissibles.

Cependant une méthode de guidage arrive à relier n'importe quelle configuration à partir d'une configuration initiale (si cette méthode ne vérifiait pas cette propriété, la voiture ne serait pas contrôlable).

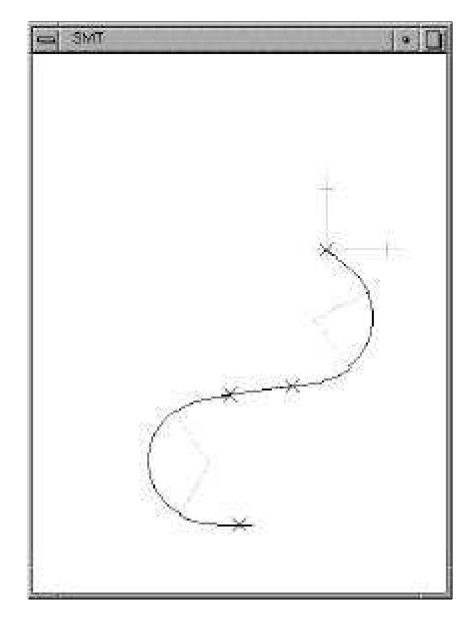
4. Propriétés du chemin CC

Nous examinons ici quelles sont les conséquences des contraintes spécifiées initialement (continuité de la courbure notamment) sur les méthodes de guidage.
On définit ici 'chemin CC' un chemin qui a pour propriété d'avoir une courbure continue (Continuous Curvature). Pour un véhicule se déplaçant uniquement en marche avant, la nature du chemin est facile à établir.

La continuité de la courbure impose que lorsqu'un véhicule amorce un virage, il ne peut pas immédiatement décrire un arc de cercle, ce qui signifierait que la courbure passe de la valeur 0 à la valeur κ_{max}, ,donc qu'elle soit discontinue. La courbure variant de manière continue, il faut modéliser cette variation. On modélise la variation de la courbure comme linéaire, pour simplifier, mais en étant tout de même très proche de la réalité. Une courbe ayant la propriété d'avoir un rayon de courbure variant de façon linéaire s'appelle une clothoïde. Un chemin CC est donc composé de

- Segments de droite
- Arcs de cercle
- Arcs de clothoïde

Lorsque le véhicule tourne, il commence donc par braquer les roues et suivre un arc de clothoïde, puis lorsque la courbure atteint la valeur κ_{max}, il décrit un arc de cercle, puis redresse les roues en suivant un autre arc de clothoïde. Lorsque la valeur de la courbure est nulle, il décrit à nouveau un segment de droite.



Chemin de type TST (virage, ligne droite, virage).
Les virages sont composés de deux arcs de clothoïde symétriques et d'un arc de cercle (les traits fins sont les rayons de ce cercle).

On peut encore définir un objet important pour les chemins CC : le **CC-Circle**. Lorsque le véhicule tourne, il ne décrit pas exactement un arc de cercle mais un virage composé d'un arc de clothoïde (correspondant au braquage des roues), d'un arc de cercle (correspondant à la partie du virage sont braquées au maximum), puis d'un deuxième arc de clothoïde symétrique (correspondant au redressement des roues). Les

arcs de clothoïde sont toujours de même longueur et l'angle du virage est déterminé par la longueur de l'arc de cercle (ou « l'angle » de l'arc de cercle). Si on appelle Ω le centre de l'arc de cercle du virage, on définit le CC-Circle comme le cercle de centre Ω et passant par la configuration initiale du virage et la configuration finale du virage (il a donc un rayon légèrement supérieur au rayon du cercle que décrit le véhicule et le virage est strictement inclus dans le CC-Circle).

5. Les différentes méthodes de guidage

A partir de cette description, on peut définir méthodes de guidage calculant des chemins CC et définir ainsi plusieurs méthodes de guidage.

- La méthode de guidage '*CC-Dubins*' (du nom de la personne ayant déterminé la forme des plus courts chemins pour la voiture « marche avant ») calcule des chemins à courbure continue et où le véhicule se déplace uniquement en marche avant. Pour rejoindre deux configurations, on peut soit :
  i. faire un virage, une ligne droite, puis un virage.
  ii. faire un virage, un virage dans le sens inverse, puis un virage dans le sens initial.

  On peut démontrer qu'il y a 6 sortes de chemins qui minimisent la longueur parcourue entre deux configurations : LSL, LSR, RSL, RSR, LRL, RLR (S représente une ligne droite, L un virage à gauche, R un virage à droite). Il faut d'abord tester l'existence de chaque sorte de chemin puis le calculer s'il existe. S'il y a plusieurs chemins possibles, le chemin le plus court détermine le chemin CC-Dubins reliant les deux configurations.

- La méthode de guidage *'CC-Reeds and Shepp'* (du nom des personnes ayant déterminé la forme des plus courts chemins pour la voiture «marche arrière »). Cette méthode de guidage calcule des chemins pour des véhicules ayant la capacité d'aller en marche arrière. On peut ainsi faire beaucoup plus de sortes de mouvements. Reeds et Shepp ont défini à l'origine des familles pour les chemins à courbure non continue. On reprend les mêmes familles pour les chemins à courbure continue. Cette méthode calcule donc le chemin le plus court parmi les neuf suivantes :
  i. $l^+l^-l^+$ ou $r^+r^-r^+$
  ii. AcAA
  iii. AAcA
  iv. AAcAA
  v. AcAAcA
  vi. AcASAcA
  vii. AcASA
  viii. ASAcA
  ix. ASA

  A représente un virage, S une ligne droite, et c un point de rebroussement.
  De même que pour le chemin CC-Dubins, il n'existe pas toujours neuf chemins possibles entre deux configurations, il faut donc d'abord tester l'existence d'un chemin avant de le calculer.

- La méthode de guidage CC-Dubins calculait un chemin parmi 6 familles données et le chemin le plus court se trouvait parmi ces familles. Cependant le chemin le plus court ne se trouve pas parmi les neuf familles précédentes. D'où l'idée d'étendre la famille à un nombre plus important. On a donc défini une méthode de guidage *'Reeds and Shepp' étendue*

- La méthode de guidage *'Reeds and Shepp' étendue*
  Elle calcule un chemin parmi un nombre de familles plus importants: la famille étendue se compose de quarante familles incluant les neuf familles précédentes. se composent des familles suivantes :

(i)-(iv)	T[l] S[l]T	
(v)-(viii)	T[l]T[l]T	
(ix)-(xvi)	T[l]T[l]S[l]T	où [l] indique un changement de
(xvii)-(xxiv)	T[l]S[l]T[l]T	direction optionnel.
(xxv)-(xl)	T[l]T[l]S[l]T[l]T	

Le temps de calcul de ces familles est plus long car le nombre de familles est plus important mais les chemins sont plus complexes donc aussi plus longs à calculer. On remarquera cependant (cf. la suite de ce rapport) que la longueur des chemins est parfois beaucoup plus courte. Le chemin le plus court n'appartient pas non plus à la famille *'Reeds and Shepp' étendue* (plus exactement, il existe des cas où le chemin le plus court n'appartient pas à cette famille, contrairement à la famille CC-Dubins).

- La méthode de guidage *topologique* .
  Les méthodes de guidage calculent un chemin en l'absence d'obstacles et sont utilisées à l'intérieur d'autres algorithmes de planification de mouvement qui eux, permettent l'évitement d'obstacles. Cependant, pour pouvoir être insérée dans l'algorithme de planification de mouvement, la méthode de guidage a besoin de vérifier certaines propriétés :

$$\forall \varepsilon>0, \exists \eta>0, \forall (q_1,q_2) \in C^2, q2 \in B(q_1,\eta) \Rightarrow Steer(q_1,q_2) \subset B(q_1,\varepsilon)$$

(C est l'ensemble des configurations, B dénote une boule fermée, Steer le chemin calculé par la méthode de guidage entre les configurations q1 et q2).

Ceci signifie d'un point de vue topologique que :
dans toute boule (ici un cercle car on est dans le plan) de rayon $\varepsilon>0$, il existe un $\eta>0$ pour lequel on doit pouvoir réunir par un chemin inclus dans cette boule deux configurations distantes de cet η. La longueur du chemin topologique n'est donc pas toujours supérieure à une constant donnée, ce que les méthodes de guidage précédentes ne vérifiaient pas cette proposition puisque la longueur des chemins était toujours au moins égale à $2*r_{cc}*\mu$.
Le détail des chemins topologiques sera exposé dans la partie 'Travail effectué'.

IV. Travail effectué

Pour mieux comprendre mon travail, je vais d'abord présenter le travail qui avait déjà était effectué pour avoir une vue globale du projet puis je présenterai plus précisément ce que j'ai personnellement ajouté ou complété. J'ai effectué ce travail seul (pas en binôme) et j'avais à ma disposition les moyens informatiques nécessaires.

1. Ce qui existait déjà

Les méthodes de guidage pour le calcul des chemins *CC-Dubins* et *Reeds and Shepp* existaient déjà. Ces algorithmes étaient testés et validés. Une interface graphique et une fenêtre de commande associée étaient également implantées, avec le logiciel FLTK. L'interface graphique permet de tester très facilement les méthodes de guidage : lorsqu'on clique sur l'interface, on peut définir des positions de configurations et ainsi tracer le chemin correspondant.

La plupart des spécifications était donc fournie au départ (*i.e.* la spécification des types Chemin, Configuration,...). Ces algorithmes sont écrits dans le langage C++.

2. Le travail effectué

Le début de mon stage a été consacré à l'étude de la partie théorique du stage, à la compréhension de l'algorithme existant, ainsi qu'à l'étude du langage C++ pour mieux me familiariser avec la programmation orientée objets. Il a donc fallu que je lise et comprenne la documentation existante sur la théorie de la planification de mouvement. Ensuite mon stage a été découpé en quatre tâches principales : l'extension de la méthode de guidage '*Reeds and Shepp*' (cf. partie théorique), la spécification des chemins topologiques et l'implantation de la méthode de guidage topologique, l'implantation de fonctions permettant de faire des statistiques et des comparaisons sur les différentes méthodes de guidage, et le développement d'une interface graphique.

i. L'extension de '*Reeds and Shepp*'

Comme nous l'avons vu dans l'explication théorique du sujet du stage, la méthode de guidage '*Reeds and Shepp*' possède une extension de neuf à quarante familles de chemin. Il faut donc implanter les familles restantes. Rappelons que la méthode de guidage '*Reeds and Shepp*' calcule les quarante chemins associés à chaque famille puis retourne le plus court chemin parmi ceux-ci. Pour chaque famille de chemin, le calcul du chemin entre deux configurations se fait de la façon suivante : test de l'existence du chemin, puis calcul du chemin s'il existe. Un chemin d'un type donné n'existe pas systématiquement, mais le chemin calculé par la méthode de guidage existe toujours (les quarante chemins entre les deux configurations n'existent pas tous, mais au moins un dans la famille existe, quelque soit le couple de configurations). On peut légitimement se demander pourquoi un chemin n'existerait pas entre deux configurations. Prenons l'exemple de la famille TT.

Le chemin TT est composé de 2 virages. Les deux CC-Circle doivent donc être tangents (j'ai brièvement défini le CC-Circle dans le sujet du stage).
En notant r_{cc} le rayon du CC-Circle, on remarque que la distance entre les deux configurations doit être exactement égale à 2*rcc pour que le chemin puisse exister. Il faut aussi examiner le sens du virage. Le chemin ne peut pas exister si les deux virages se font dans le même sens :le deuxième virage doit être de sens opposé au premier virage.
Si le chemin existe, il faut ensuite le calculer. Le but n'étant pas d'entrer dans les détails de l'algorithme, je présenterai simplement la méthode pour calculer un chemin.

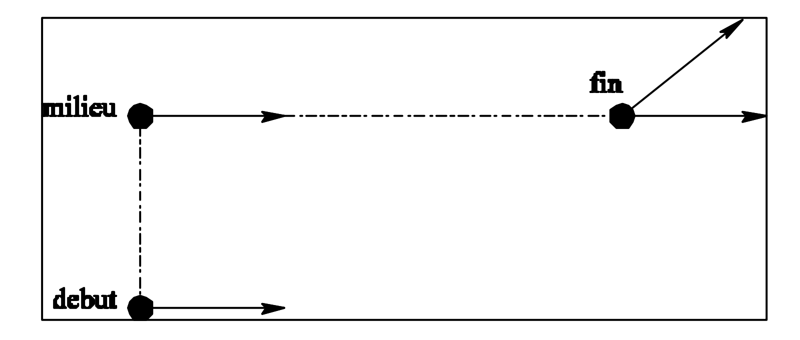
Le but est de calculer un chemin '*Reeds and Shepp*' (ou '*Reeds and Shepp* étendu') entre deux configurations. Pour cela, on calcule donc les quarante chemins de la famille '*Reeds and Shepp*' et on choisit le plus court. Pour calculer un chemin d'une famille donnée (par exemple TT), on procède de la manière suivante. Pour chaque configuration, il y a 4 CC-Circle associés (on peut aller à gauche ou à droite, et en marche avant ou arrière). Pour le couple de configurations initiale/finale, il y a donc 16 couples de CC-Circle différents. Pour chacun de ces couples, une fonction teste l'existence d'un chemin et le calcule s'il existe (il peut y avoir plusieurs chemins existants, un seul, ou aucun ; cela dépend du type du chemin et de la position des configurations).
Par exemple, pour le chemin TST (virage, ligne droite, virage), il y a plusieurs chemins possibles : les deux virages peuvent être dans le même sens ou dans un sens opposé. Le plus court sera différent selon les cas.
Pour le calculer, il faut calculer les configurations intermédiaires et les CC-Circle intermédiaires du chemin (pour pouvoir ensuite tracer le chemin et calculer sa longueur). Plus le chemin est complexe, plus il y a de configurations intermédiaires à calculer (ce qui est le cas pour la famille étendue...) et d'éventuels chemins possibles.

Il est facile de tester la validité de l'algorithme de calcul des chemins : un outil graphique était déjà à ma disposition pour visualiser les chemins. On peut alors facilement vérifier si le calcul du chemin est correct.

ii. Calcul du chemin topologique

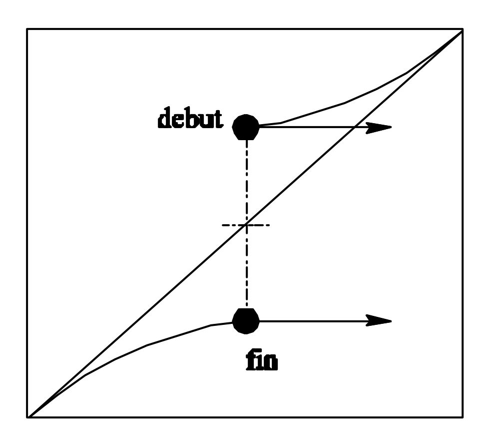
Comme je l'ai brièvement rappelé dans la présentation de mon stage, le chemin topologique a été implanté pour vérifier une propriété topologique, ce que les deux méthodes de guidage précédentes ne font pas. Le chemin topologique permet aussi, nous le verrons dans la suite du rapport, de réduire considérablement la taille des chemins pour des configurations voisines.

Le chemin topologique se compose de trois chemins : un chemin latéral, un chemin en ligne droite, et un chemin de réorientation.



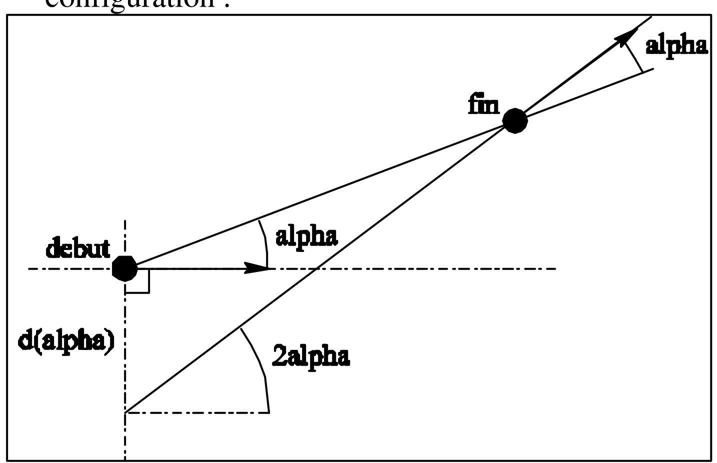
Le chemin topologique relie deux configurations initiale et finale ('*debut*' et '*fin*') via une configuration intermédiaire ('*milieu*').

- Le chemin latéral relie deux configurations qui ont une orientation identique. Il sert à se positionner «en face » de la configuration finale (il faut faire une ligne droite pour parvenir à la position de la configuration finale). Il est composé de trois mouvements :
  1. un virage en marche avant
  2. une ligne droite en marche arrière
  3. un virage en marche avant.



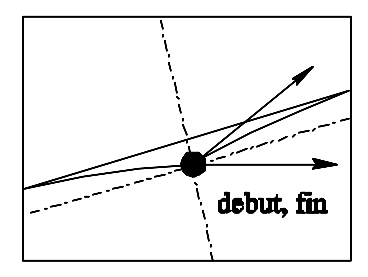
Chemin topologique entre les configurations 'debut' et 'fin'.

Comme il existe une infinité de mouvements latéraux entre deux configurations, on le contraint à être symétrique par rapport au milieu du segment reliant les deux configurations. Pour déterminer les configurations intermédiaires du chemin, il faut trouver l'angle de braquage des roues (alpha) nécessaire pour rejoindre la deuxième configuration .



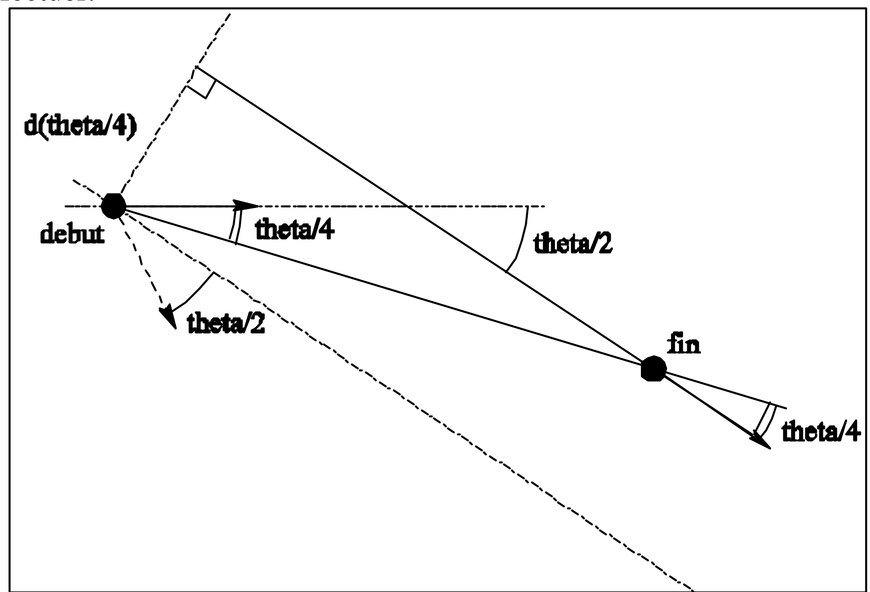
L'angle alpha dépend de la distance séparant les configurations et correspond à la valeur du 'braquage' des roues.

- Le troisième mouvement (mouvement de réorientation) a un principe identique à celui du mouvement latéral. Il sert à redresser les roues, une fois que l'on a atteint la position de la configuration finale .
  Il est composé de trois mouvements :
  1. un virage en marche arrière
  2. une ligne droite en marche arrière
  3. un virage en marche avant.



Le mouvement de réorientation sert à redresser les roues du véhicule pour aller dans la direction souhaitée.

Là aussi, il existe un infinité de mouvements de réorientation : c'est pourquoi on contraint ce mouvement à être symétrique par rapport à une droite remarquable (cf. figure). Suivant l'angle de redressement des roues, on détermine l'angle de braquage nécessaire à effectuer.



Si l'on veut redresser les roues d'un angle theta,
il faut braquer les roues d'un angle theta/4.

Une fois le chemin topologique implanté (définition de la classe, calcul des chemins, puis test), on implante une « super » méthode de guidage qui inclut à la fois le chemin topologique et le chemin '*Reeds and Shepp*' étendu.

iii. Statistiques et comparaison.

Une fois ces méthodes de guidage implantées, il est intéressant de savoir quelle méthode est la plus efficace et de quantifier les optimisations successives sur les méthodes de guidage (‘*Reeds and Shepp*’, ‘*Reeds and Shepp*’ étendu, chemin topologique). Pour cela on définit une fonction de statistiques et une fonction de comparaison qui permettent de comparer deux méthodes de guidage. La fonction de statistiques calcule un nombre donné de configurations finales de manière aléatoire, pour qu’elles soient réparties de manière la plus uniforme possible, et calcule les deux chemins correspondants, en utilisant les méthodes de guidage *ad hoc*. Le nombre de chemins doit être grand pour avoir une haute précision (mais le temps de calcul peut être alors un peu long : quelques minutes pour comparer 50 000 chemins). Cette fonction calcule :

- la longueur du plus court chemin et du plus long pour les deux méthodes de guidage
- les moyennes des longueurs des chemins
- les rapports de ces longueurs
- l’écart type (pour savoir si les différences entre les longueurs de chemins sont régulières ou pas)
- le temps de calcul nécessaire à chaque méthode de guidage

Il est aussi nécessaire de préciser le rayon de la zone dans laquelle on veut trouver les configurations : au-delà d’une certaine limite, toutes les familles renvoient le même chemin (le chemin le plus court est le chemin simple TST...).

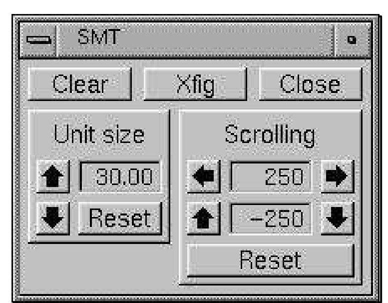
Cette fonction de statistiques est très utile car elle permet de voir les performances relatives entre les différentes méthodes de guidage.

Il est aussi possible de comparer les chemins directement à l’aide de l’interface graphique. On peut définir la position de la configuration finale (en cliquant à l’endroit voulu) et ainsi comparer avec les différentes méthodes de guidage leurs performances respectives (on fait varier la position et l’orientation du véhicule).

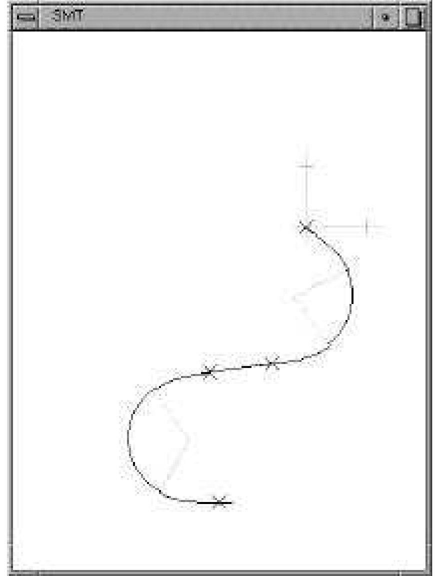
iv. Interface graphique

Pour faciliter le fonctionnement de l'interface, une autre interface a été développée. A l'origine, seule une petite fenêtre de commande existait (cf. figure). Celle-ci permettait de zoomer sur la figure, de faire un scrolling, et traçait automatiquement les chemins '*Reeds and Shepp*' et '*CC-Dubins*'. Cette interface n'est plus très pratique dès qu'il y a beaucoup plus familles de chemins implantées et que l'on désire plus de souplesse (modification directe de l'orientation des roues, choix d'un type de chemin particulier, utilisation de la fonction statistique...). La nouvelle interface est construite avec le logiciel FLTK (comme les deux fenêtres déjà existantes). Ce logiciel est assez simple d'utilisation (avec un peu d'habitude...).

L'interface graphique et les deux autres fenêtres déjà existantes figure ci-dessous.

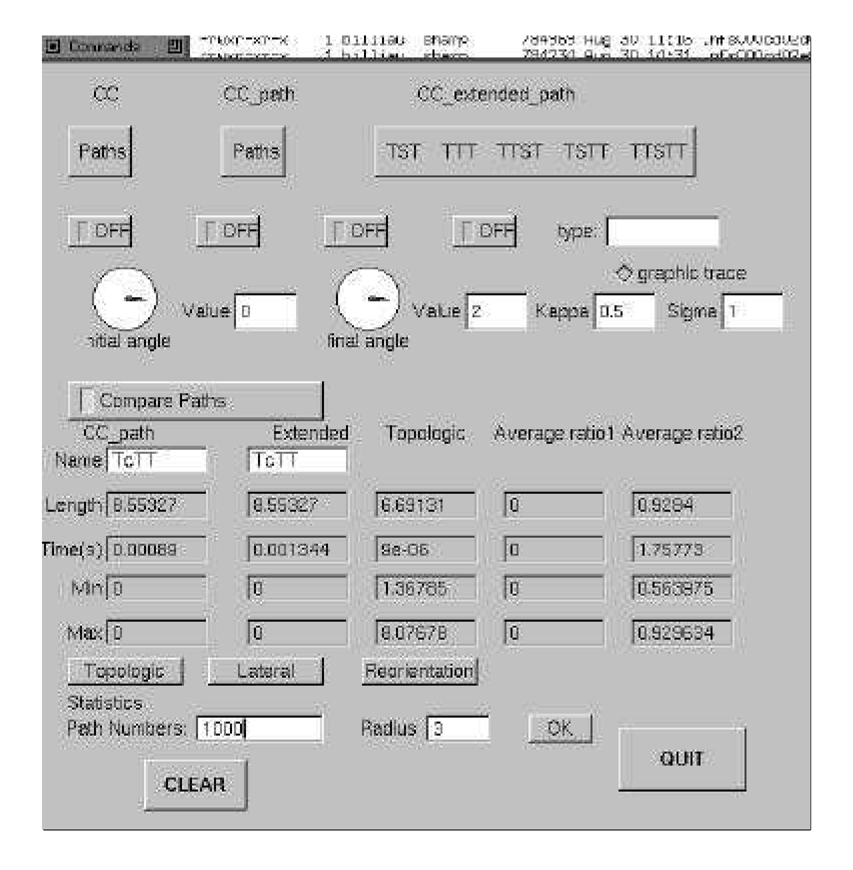


Fenêtre déjà existante permettant de faire des zooms et du scrolling



Fenêtre déjà existante permettant d'afficher les Chemins (ici chemin de type TST).

Sur l'interface créée, un menu a été de commandes a été implanté pour permettre de sélectionner un type de chemin pour une méthode de guidage donnée. On peut alors le tracer et faire des tests plus facilement. Des boutons ON/OFF permettent de sélectionner le choix du chemin à afficher. Il est aussi possible de changer l'orientation des roues (orientation initiale ou finale). Des fenêtres permettent de connaître directement la longueur du chemin tracé et le temps de calcul du chemin.



Interface créée avec FLTK.

L'utilisation de FLTK est plutôt simple : à chaque bouton est associé un callback. Plus précisément, une fonction callback est appelée à chaque fois que la valeur du widget (l'interface graphique) associé est modifiée. Il m'a donc fallu définir l'interface et les boutons, et définir aussi tous les callback associés. La création de cette interface m'a pris un certain temps, car je l'ai modifiée en même temps que je modifiais le programme principal, car alors certains boutons n'avaient plus lieu d'être ou pouvaient être rassemblés en un seul.

Le reste du temps de mon stage a été consacré à des tests et au débuggage du code (ou à son optimisation) et aussi à la création d'une documentation en ligne.

v. Documentation

J'ai complété une documentation détaillée sur le programme avec l'outil Doxygen. Cette documentation contient la présentation du programme, la définition détaillée et la hiérarchie des classes, la liste des codes source commentés, un index de toutes les fonctions et des attributs des classes. Cette documentation est une documentation html, facilement consultable en ligne, et il est possible de générer la documentation au format Latex pour avoir une documentation papier totalement complète (environ 200 pages). Cette documentation doit être structurée pour être facilement utilisable. C'est une documentation technique et aussi un manuel d'utilisateur : elle détaille certains points techniques des algorithmes et explique aussi comment utiliser l'interface. Les points de détail dans les algorithmes ne sont pas mentionnés, pour laisser en relief les points importants du programme. Il faut donc veiller à ne rien mettre de superflu, et expliquer les points algorithmiques demandant un minimum d'explication.

IV. Conclusion. Bilan global

J'ai trouvé ce stage intéressant dans la mesure où il m'a permis d'aborder un sujet sur lequel je n'avais jamais travaillé : la robotique. Il était pour moi très motivant de travailler sur une partie d'un projet mobilisant beaucoup de monde et qui avait un résultat plutôt spectaculaire : la construction de véhicules autonomes. Il m'a aussi permis de compléter mes connaissances en programmation orientée objet et plus généralement en algorithmique. Il m'a aussi permis de voir comment se passait le travail en laboratoire, dans le projet Sharp mais aussi plus généralement dans les autres projets de recherche à l'INRIA, en étant en contact avec d'autres étudiants en stage dans d'autres projets. Je regrette de n'avoir pas pu aller plus loin dans mon stage, c'est-à-dire tester l'algorithme sur des véhicules autonomes, même s'il y avait encore beaucoup de tests à mener.

Bibliographie

[1]. Th. Fraichard et J.-M. Ahuactzin. *Smooth Path Planning for Cars*. IEEE Int. Conf. On Robotics and Automation, May 2001.

[2]. Th. Fraichard, A. Scheuer et R. Desvigne. *From Reeds and Shepp's to Continous-Curvature Paths*. IEEE Int. Conf. On Advanced Robotics, October 1999.

[3]. Th. Fraichard. *Chemins à courbure continue*. INRIA Rhône-Alpes, 8 août 2001.

Index des mots-clés.